Using CNNs to Identify Events in Basketball Videos

Andrew Chang
Stanford University
achang97@stanford.edu

Akshay Ramaswamy
Stanford University
aramaswa@stanford.edu

Chetan Rane
Stanford University
chetankr@stanford.edu

Abstract

Classification of actions in basketball games is an important problem to solve, since sports analysts spend a disproportionate amount of time manually looking through basketball highlights in order to produce relevant commentary and statistics. In this paper, we implement several different convolutional neural network models to identify actions happening on a basketball court, using a dataset of 14,000 NCAA basketball clips. We tried a range of architectures, from a simple baseline CNN trained from scratch to more complex 3D convolutional nets to capture temporal data. Our most successful architecture ended up using 3D convolution on a pre-trained Sports-1M model, achieving 35.45% test accuracy and 26.1% average precision over the 11 classes. This outperformed prior state of the art 3D convolution methods on classification of basketball actions, which achieved an average precision of 22.1%.

1. Introduction

Every day, sports networks such as ESPN, FOX, ABC, and CBS release hundreds of sports analytics reports for the public to view. While these reports capture precise play-by-play analytics and overall game statistics, this is only accomplished through the written records of sports analysts that spend hours watching and documenting games. We believe that with the use of computer vision, we can report similar accurate play-by-play events from a sports game without the need of the human eye. In this paper, we turn our attention to the game of basketball, which involves quick ball movements and a wide variety of noticeable actions, ranging from passing to dribbling to shooting. In tracking individual player movement and classifying the action each player on the court takes, we can provide valuable insights and statistics for more advanced sports analytics.

The inputs to our algorithms include both 2D images and 2D videos, as we attempt to use both image and video classification methods. We then use a 2D convolutional network for images and a 3D convolutional network for videos to output a predicted basketball action. Predictably, using image classification techniques to classify videos performed very poorly, as doing so completely ignored the temporal relationship between related images in the same clip. This method performed not much better than random guessing. Our best results came from using 3D convolutional nets, which achieved results more comparable to state-of-the-art methods.

2. Related Work

After assessing recent papers on action recognition from video data, it is apparent that the task is well-studied but rather difficult for a multitude of reasons. State-of-the-art models still present classification accuracies below 50%. Nonetheless, tremendous improvements have been made by applying deep learning to the task of action recognition.

i. Action Detection Datasets

There exist several datasets for action classification in video samples. These datasets range from detecting legitimate motions in sports videos as apparent in Sports-1M [1], UCF-Sports [2], THUMOS [3], etc. to recognizing general human activity in datasets such as Hollywood [4],

MSR Action [5], ASLAN [6]. Such datasets are commonly used for pretraining models to understand generic human actions, such as throwing and running, which oftentimes also appear in many sports clips. The difficulties regarding action recognition in sports videos stem from challenges with generating and labeling enough data for classification and the variability of different actions across classes [7]. One of the strengths of the increasing number of action recognition datasets published for public access is its utility in constructing pretrained models to assist with more complex action classification tasks. Sports-1M is a very popular and comprehensive dataset, making it an excellent source for pretrained models.

ii. Action Recognition in Videos

Various models have been applied towards the task of action recognition in videos; however, the performance of most of these methods is still beneath the capability of human vision. In the past few years, deep learning has notably outperformed feature-based classifiers on the task [8, 9, 10]. Karpathy et al. [8] was the first to train a deep convolutional neural network on the Sports-1M dataset, which contains over 1.1 million videos. Such deep learning approaches present clever solutions to a rather difficult problem by discovering new features rather than engineering them. The strengths of these approaches include a significantly better performance than HOG gradients and feature extraction, and thus, feature determination from conv nets appears to be a well-performing model employed in most action recognition tasks. One of the downsides of video representations based on spatio-temporal convolutions is the difficulty presented in scaling to longer videos when attempting to learn over the full video. [11, 12] On the other hand, RNNs have obtained state-of-the-art results in video-based action recognition [13, 14, 15], and these have been increasingly appealing due to their ability to map variable length inputs to variable-length outputs. Thus, they have been used for generating captions and could be applied for live commentary, a real-time application of basketball action recognition in live clips.

Our conclusion from past work in action recognition from video data is to proceed by using RGB values to characterize the data. For this paper, we will employ a CNN to make use of spatio-temporal features. Finally, we'll utilize the Sports-1M dataset to pretrain our model. Several papers have focused their efforts on training RNNs, but we believe that we can produce competitive results using CNNs and well-structured data due to the development of more recent models [18].

3. Methods

Using the various methods we learned about in class and in the papers we read, we tried out a multitude of architectures for our models. Our first set of models were based on simple image classification methods. These methods did not make use of temporal data and were trained from scratch. The second set of models made use of temporal data, as we tried using 3D convolution. Finally, rather than starting from scratch, we made use of transfer learning and used models pre-trained on the Sports-1M dataset.

For all approaches, we utilized the softmax function as our measure of loss, as it yielded an intuitive interpretation of probabilities for each class.

Eq. 1 $$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

i. Simple Baseline

First, rather than attempting to classify the entire clip, we examined each individual frame and matched it to the corresponding label of its encompassing clip. We then classified individual images using several different CNNs. Our first

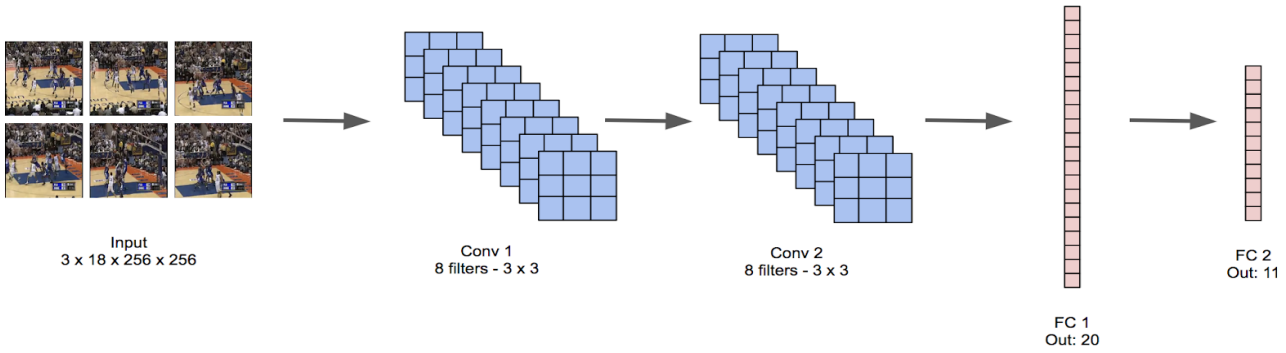

Figure 1. Baseline architecture for a 3D convolutional neural net

architecture was a simple two-layer net, just to make sure we were getting accuracies and losses better than random. To better learn the features of the images, we then implemented deeper net architectures, beginning with the standard Conv-ReLU-Conv-ReLU-FC architecture from class. The convolution layer operates by convoluting an activate filter over the image and computing the dot products, with the benefit of preserving the spatial structure of the image rather than flattening it out. With each activation filter connecting to a local set of neurons, we ideally can detect higher level features, like the relationship of the basketball with the coordinates of each individual player. The last fully connected layer connects to the entire input volume, enabling us to convert the convolution operations into classification scores for each class. We used the ReLU activation because it does not saturate in the positive region, is computationally efficient, and converges faster than sigmoid or tanh. We tried tweaking this general architecture with pooling and batch normalization as well, which interestingly did not make much of a difference.

ii. Majority Vote

Our next approach attempted to make use of the vital temporal data, but not to the extent that a 3D convolution net would. We still classified individual images, but we took a majority vote of the classifications for a single image to produce a single classification for an entire clip. While far from an optimal model, this approach addressed the problem of forcing a classification through extremely limited data, as we observed in single-frame classification. Rather, we pooled together the classifications of each 20-image clip to generate a more accurate prediction.

iii. 3D Conv Net

To better capture temporal information, we made use of a 3D convolutional net with a time dimension. In class, we learned about 2D convolutional nets, which we went into detail in our baseline. In the 3D model, we take these 2D images and stack them on top of each other (channels x time x height x width). We then apply 3D activation filters to preserve the spatial structure in all dimensions, sliding the filter across the spatial dimensions. These convolutions compute dot products between the filter and the input tensor over the 3D intersection as opposed to the 2D intersection, sliding over the H, W, and T dimensions.

The structure of our convolutional net was Conv-ReLU-Conv-ReLU-FC- ReLU-FC. Our primary architecture used 3x3 filters, because a smaller filter size uses fewer parameters while achieving the same effect. We only had 8 filters however, because our cloud instance didn't have enough RAM to use any more filters. The spatial dimensions were 18x256x256, and after each convolution layer we used a ReLU activation function. We used 256x256 because this was the largest power of two we could resize from our initial images (we later resized to 160x160 to use

Figure 2. Selected frames from an example video of 3-point success, resized to 160 x 160

our pretrained model, explained in transfer learning below). We flattened these dimensions to run the images through our fully connected layers, which we then used to create scores for the 11 classes (see **Figure 1**).

v. Transfer Learning

Running the 3D convolutional net from scratch produced poor loss and accuracy results, so we decided to try pretraining the model on the Sports-1M dataset. The training of a very deep 3D CNN from scratch also results in expensive computational cost and memory demand. We chose the P3D model because it was one of the highest performing models on the Sports-1M dataset, with an accuracy close to 90% (better than state of the art), and because it could be applied to 3D models - meaning that we wouldn't have to later train on each 2D image individually, but could rather train on the entire clip and maintain the temporal structure. The structure of the P3D model consisted of a series of 3D convolutional layers, 3D BatchNorm layers to normalize the data, ReLU activation functions, and max pooling, with the last few layers utilizing average pooling and dropout. The P3D model also made use of parallel filters and bottlenecking to maintain a reasonable feature depth, similar to state-of-the-art GoogLeNet / Inception. We replaced the very last fully connected layer with our model. Using the weights from the pretrained model, we trained the 3D convolutional net on our actual dataset, which ended up dramatically improving our accuracy. Lastly, we finetuned the last two layers of the network, and this made a big difference in our accuracy as well.

4. Dataset and Features

Initially, we used the dataset provided in this sports analytics challenge from the University of Toronto. This dataset provides 550 "videos" of 257 NCAA games, in the form of directories of images, even distributed over 11 different actions: other 2-pointer success, steal success, other 2-pointer failure, slam dunk failure, slam dunk success, layup success, free-throw success, 3-pointer success, 3-pointer failure, free-throw failure, and layup failure. Each video contains up to 20 frames and varied in resolution, with the majority having dimensions of 360 x 490 and others having dimensions of 720 x 1280. We decided to use 80% of the data for training, 5% for validation, and 15% for testing, which resulted in approximately 440, 27, and 83 examples for each category respectively. In order to use pre-trained models such as Inception for images and P3D199 for videos, we resized the images spatially to 299 x 299 and 160 x 160. We also normalized the images along height and width for images. We also manually created videos from the provided images by concatenating the images along an additional temporal dimension and normalized over time, width, and height. This dataset provides no information about the frame rate of the images and thus we had little intuition about the

discretization of the time-series apart from the number of images or frames for each clip.

However, this size of the dataset led to poor performance on various models. We decided to create our own dataset utilizing the information used in Vignesh Ramanathan et. al's paper [9]. This is a superset of the previous Toronto dataset and contains the same 11 output actions, but provides over 14,000 videos that can be used. Since the dataset provides only YouTube video ids, we wrote a script utilizing ffmpeg and youtube-dl to scrape the videos, resize each video to 160 x 160 pixels, and standardize each clip to be 3 seconds long with a frame rate of 6 FPS (resulting in 18 frames for each video). Due to memory and throughput limitations, we used a subset of 10406 examples and the same percentage splits as before for training, validation, and testing (resulting in 8309, 523, and 1574 examples respectively). We perform normalization as with the previous dataset. One downside of this dataset is its inclusion of unequal amounts of each action, leading to class imbalance.

We did not explicitly extract features, instead utilizing the RGB pixel values and allowing the convolutional layers in our neural network to identify important features of the dataset.

5. Experiments, Results, Discussion

i. Experiments

a. Hyperparameters

For each method, we used a standard set of hyperparameters to tune our model on the validation set. Because our best approach utilized a pretrained model with predetermined layers and weights, we focused mainly on our learning rate, which we chose for gradient descent through random selection over a log scale.

We also used the Adam optimizer in order to adaptively tune the learning rates per parameter, which utilizes momentum to converge towards the optimal learning rate faster. We ultimately implemented this with a weight decay of 1e-8 for optimization, though we also experimented with several random values. The other aspects of our experiments were impacted heavily by our limited RAM and the slow speed of reading mp4 files, which limited our maximum minibatch size. On a related note, we decided against using cross validation, as this approach would have resulted in an unrealistic training runtime.

b. Primary Metrics

Our main evaluation metric was the accuracy of our model on the overall test set. However, we also looked at recall and precision for each individual class in order to measure the performance of our model on different actions. We wanted to look at these metrics due to the imbalanced dataset, though we ultimately wanted to maximize the overall test accuracy.

ii. Results

The results from our initial approach using image classification were predictably poor, performing only slightly better than a random classifier.

Figure 3 shows the loss results from single image classification. There is no real trend in the loss, likely due to the fact that individual images from different classes look remarkably similar. The differences lie in the temporal dimension of the videos, which cannot be captured using this baseline approach. Tuning the learning rates had no effect, and the predictions remained not much better than random (which we would expect to be around 1 / 11 = 9.09%). The accuracies for this model, shown in **Figure 4,** maxed out at around 11% for the validation set and test set.

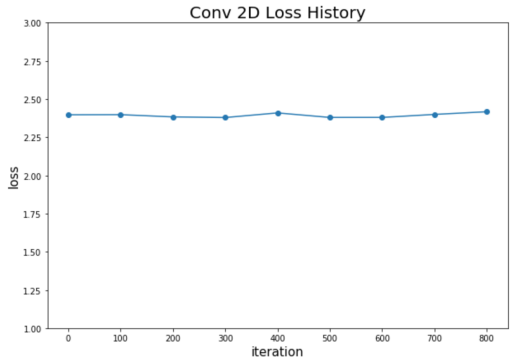

Figure 3. Loss history using image classification

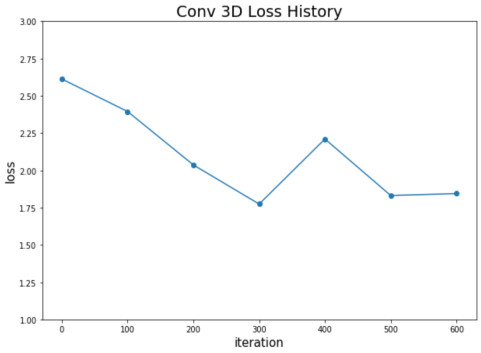

Figure 5. Loss history of pretrained 3D ConvNet

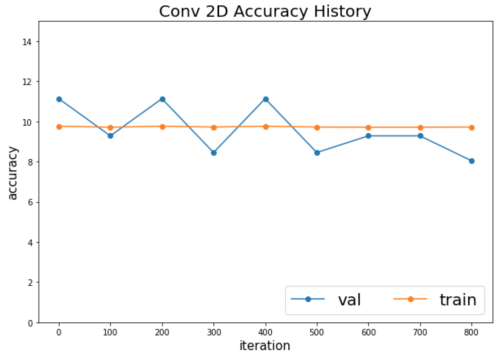

Figure 4. Accuracy history using image classification

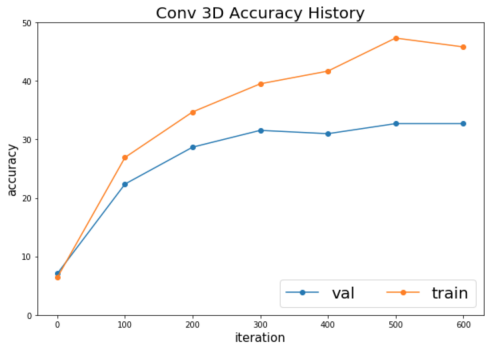

Figure 6. Accuracy history of pretrained 3D ConvNet

The majority vote model, which classified an entire video using the classifications of its individual images, still only performed marginally better than our initial approach, peaking at an accuracy of around 14%.

The next approach utilized the temporal dimension more effectively, as we constructed a 3D convolutional network with a temporal dimension that could more effectively determine important features of the data. Surprisingly, without transfer learning, the model performed worse than the one utilizing majority vote, maxing out at a test accuracy of 13%.

However, using a pretrained P3D-199 ResNet Model [18] yielded results exceeding those of state-of-the-art 3D ConvNets [9].

The loss curve in **Figure 5** is much more encouraging, decreasing at almost checkpoint except at iteration 400. The accuracy curves in **Figure 6** for both validation increase nearly monotonically.

iii. Discussion

As is shown in **Figure 7**, the model that utilized transfer learning dramatically outperformed the models trained from scratch. Among our models trained from scratch, the majority vote model performed the best, followed by 3D convolution trained from scratch. Our 2D convolutional net performed the worst, producing an accuracy essentially close to random.

The biggest reason why our 3D convolutional net with transfer learning ended up achieving better than state-of-the-art accuracy was because we pretrained it on the Sports-1M dataset. This enabled it to effectively detect edges, corners, and other important attributes related to sports (like where the ball is in the court, or where the people

Method Name	Test Accuracy
2D ConvNet Baseline	9%
Majority Vote	12%
3D ConvNet	13%
3D ConvNet with Transfer Learning	35.45%

Figure 7. Test accuracies of each attempted model

are standing on the court), which thus gave it the ability to able to output very useful features. The issue with our over-simplified baseline and 2D models is the majority of images examined in the training data are very similar and thus cannot be distinguished when predicting a specific label. Unlike most classification problems, where the distinct classes have vastly differing features, all our images are shot from similar angles and have the same background of a basketball court. An image might contain a picture of a player shooting a ball, but this only occupies a small part of the overall data sample. Overall, without considering temporal information and the other images in the same clip that show the ball's trajectory, it's very hard to predict the type of action.

In addition to the test accuracy on the overall dataset, we also wanted to identify the performance of our best model on the individual classes. To do so, we created a confusion matrix (**Figure 9**) and calculated the precision and recall for each output label (**Figure 8**). Discounting the classes with lower frequencies, the model performed best on free-throw success, steal success, 3-pointer failure, and other 2-pointer failure. The success of our model on steals can be explained intuitively, as it is the only class that is not associated with a type of "shot." Our model's impressive performance on free-throw success also makes sense, considering that players must be arranged in certain locations during free throws. Therefore, these videos are likely to be much more consistent than for other categories, allowing the model to more effectively classify sequences of this type.

Interestingly, the model performed well on free-throw success but had extremely low numbers for free-throw failure. One reason for this discrepancy may be due to the fact that videos in these two categories differ only in their last few frames. A more likely source of this discrepancy is the imbalance of the dataset. We utilized cross-entropy / softmax loss to train our model without weighting each class differently, attempting to maximize the overall accuracy. Since there were more occurrences of free-throw success than free-throw failure, our model was more inclined to predict the former.

Action	Recall	Precision
steal success	**198 / 262 = 75.57%**	**198 / 372 = 53.23%**
other 2-pointer success	1 / 151 = 0.66%	1 / 7 = 14.29 %
other 2-pointer failure	**129 / 291 = 44.33%**	**129 / 445 = 28.99%**
slam dunk failure	0 / 8 = 0.00%	0 / 0 = 0%
slam dunk success	0 / 33 = 0.00%	0 / 0 = 0%
layup success	17 / 152 = 11.18%	17 / 77 = 22.08%
free-throw success	**53 / 65 = 81.54%**	**53 / 102 = 51.96%**
3-pointer success	2 / 146 = 1.37%	2 / 5 = 40.00%
3-pointer failure	**139 / 260 = 53.46%**	**139 / 464 = 29.96%**
free-throw failure	2 / 41 = 4.88%	2 / 7 = 28.57%
layup failure	17 / 165 = 10.30%	17 / 95 = 17.89%

Figure 8. Individual class precision and recall

Actual

	A	B	C	D	E	F	G	H	I	J	K
A	13	198	43	3	7	32	6	10	28	3	29
B	1	0	2	0	0	3	0	0	0	0	1
C	64	23	129	1	9	49	1	36	77	2	54
D	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0
F	14	8	19	1	3	17	0	6	5	2	11
G	3	3	3	0	1	1	53	1	3	32	2
H	0	1	0	0	0	0	0	2	1	0	1
I	45	22	77	3	11	30	2	87	139	0	48
J	0	1	0	0	0	0	2	0	0	2	2
K	11	6	27	0	2	20	1	4	7	0	17

Letter	Action
A	steal success
B	other 2-pointer success
C	other 2-pointer failure
D	slam dunk failure
E	slam dunk success
F	layup success
G	free-throw success
H	3-pointer success
I	3-pointer failure
J	free-throw failure
K	layup failure

Figure 9. Confusion matrix

To visually analyze why certain accuracies were so high, we produced a series of saliency maps to get an in-depth look at what our 3D ConvNet was determining as an important feature. In **Figure 10**, we outputted the saliency maps for an image classified as a free throw, as this had class had a relatively high precision and recall. The map results were promising, as the red was centered on the location of the players and the free throw shooter. This indicated that our model was doing a good job of auto segmenting players for certain classes, and that it was learning on the right features.

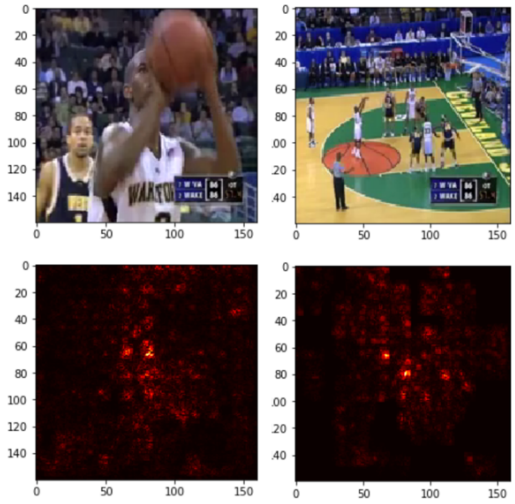

Figure 10. Saliency maps of free-throw success

6. Conclusion

i. Key Findings

The largest takeaway from this project is that pretraining on the Sports-1M dataset dramatically increases the accuracy and improves loss minimization. Our 3D ConvNet with transfer learning ended up outperforming industry standard 3D ConvNets, with an average precision of 26% and accuracy of 35.45% versus previous works achieving only 22% precision. This was most likely due to us using the P3D model model to pretrain, which has a very high accuracy on the Sports-1M dataset (~90%). Our other methods performed worse because they either didn't take into account temporal data or were trained from scratch, preventing them from learning more abstract features like the coordinates of the basketball in relationship to the other players and the court.

ii. Future Steps

With more time, we would like to train different architectures; mainly we would have liked to try running an RNN in the form of an LSTM, since several papers use this as another way to utilize temporal information. Specifically, we believe that using a CNN for feature selection to then feed into an LSTM would produce even better results, since we know our saliency maps indicate our 3D ConvNets do a good job of detecting important objects in the frame of the basketball court to use as features. We would also like to use attention for auto segmentation to identify key players on the court, rather than fully depend on the CNN to detect these players naturally. Lastly, we would have liked to use a more modern dataset with better resolution of videos, since the dataset we were able to process and obtain consisted of relatively low-resolution clips from the early 2000s.

7. Contributions & Acknowledgements

All authors collaborated to write the paper. Chetan focused on garnering information from applicable research papers and searching for useful datasets. Akshay worked on developing models and fine-tuning hyperparameters, as well as producing related graphs and charts. Andrew also worked on the models and coded the scripts to obtain and preprocess data.

We utilized the pre-trained model found in qijiezhao's GitHub repository [18] and code from various CS231N assignments [2]. Specifically, we utilized several functions from the introductory PyTorch.ipynb notebook from assignment 2 to train our models and check their accuracies. We also reused code from the NetworkVisualization.ipynb notebook for our production of saliency maps. We did not work

with any non-CS231N collaborators or other outside resources.

8. References

1. J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. arXiv preprint arXiv:1411.4389, 2014
2. Li, Fei-Fei. CS231N Assignment Code, 2018.
3. R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. C. Russell. 2017. ActionVLAD: Learning spatio-temporal aggregation for action classification. CoRR Vol. abs/1704.02895 (2017).
4. Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS
5. A. Karpathy et al. Large-scale video classification with convolutional neural networks. In CVPR, 2014.
6. O. Kliper-Gross, T. Hassner, and L. Wolf. The Action Similarity Labeling Challenge. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 34(3), March 2012.
7. I. Laptev, M. Marszalek, C. Schmid, B. Rozenfeld. Learning Realistic Human Actions from Movies. In IEEE Conference on Computer Vision & Pattern Recognition, 2008.
8. W. Li, Z. Zhang, Z. Liu. Action Recognition Based on A Bag of 3D Points. IEEE International Workshop on CVPR for Human Communicative Behavior Analysis (in conjunction with CVPR2010), San Francisco, CA, June, 2010.
9. V. Ramanathan, J. Huang, S. Abu-El-Haija, A. Gorban, K. Murphy, and L. Fei-Fei. Detecting events and key actors in multi-person videos. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3043–3053, 2016.
10. K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In Advances in Neural Information Processing Systems, pages 568–576, 2014.
11. K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556, 2014
12. K. Soomro, A. Roshan Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. In CRCV-TR-12-01, 2012.
13. N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using lstms. arXiv:1502.04681, 2015.
14. D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In ICCV, 2015
15. R. Urtasun. CSC2541: Sports Analytics (Challenge 1). https://www.cs.toronto.edu/~urtasun/courses/CSC2541_Winter17/project_1.pdf
16. G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. CoRR, abs/1604.04494, 2016
17. L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville. Describing videos by exploiting temporal structure. stat, 1050:25, 2015.
18. Q. Zhao. Pseudo-3D Residual Networks, 2013, GitHub repository, https://github.com/qijiezhao/pseudo-3d-pytorch.